

ELECTRONICALLY FILED ON

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of inventor(s):

Chris Connaughton

Application No. 09/665,888

Filing Date: 20 September 2000

Title: System and Method of Analyzing an
HTML Document for Changes Such
That the Changed Areas can be
Displayed with the Formatting Intact

Confirmation No. 6813

Group Art Unit: 2176

Examiner: Gautam Sain

CUSTOMER NO. 22470

MAIL STOP AMENDMENT

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

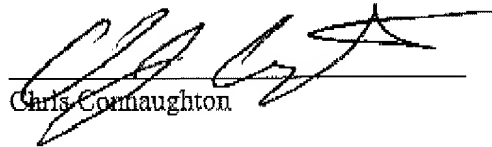
DECLARATION OF INVENTOR UNDER 37 CFR 1.131

1. I, Chris Connaughton, am inventor on the above identified patent application. I had reduced the invention set forth all claims as set forth in the RESPONSE/AMENDMENT filed 24 April 2006, within the United States before 27 August 1999.
2. The claimed invention was implemented and fully functional in the BullsEye product publicly available prior to 27 August 1999. BullsEye includes the routine which executes all the steps of claim 34. BullsEye includes the routine which executes all the steps of claim 43. BullsEye includes the routine which executes all the steps of claim 52.
3. Attached hereto is Attachment A, which contains a portion of source code of the BullsEye product showing the step of "parsing, using a computer, the first document into a first plurality of groups of characters delineated by block level markup tags". The additional elements of the claims were also implemented in the source code of the BullsEye product.
4. Attached hereto is Attachment B, which contains a redacted computer screen shot of the source code archive showing the history of the file from which Attachment A was taken from,

showing that the source code that contains Attachment A was implemented in a version dated 18 February 1999.

The undersigned declares that all statements made herein of his own knowledge are true and that all statements made on information and belief are believed to be true; and further that the statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application and any patent that may issue therefrom.

Date: 2007-1-12


Chris Connaughton

Attachment A

```
1  BOOL THtmlParser::HandleElement
2  (const string& strElement, ATTRVAL *pAttrVal[],
3  int cAttrVals, BOOL& bProceed)
4  {
5      int    i    = 0;
6      string  str;
7      ELEMENTFCNS* pEf    = NULL;
8
9      //Title tag is automatically ended if another tag begins
10     m_bInsideTitle = FALSE;
11
12     //Scan the list of element functions to see if this tag requires
13     special processing
14     for (i = 0; i < FUNCTION_ARRAY_SIZE; i++) {
15         pEf = &ElementFcnList[i];
16         //Compare tag to tag of current row
17         if (strElement.comparenocase( pEf->szElement) == 0) {
18             //Extract the "html type" associated with this tag (image, link, etc)
19             m_nLastHtmlType |= pEf->htmlType;
20             break;
21         }
22         pEf = NULL;
23     }
24     //If we're only needing to scan for html types in the input string,
25     time to leave.
26     if (m_bProcessingType) {
27         if (pEf == NULL)
28             m_nLastHtmlType |= TYPE_PLAINTAG;
29         return TRUE;
30     }
31
32     //Handle any abnormal container enders for the anchor tag
33     if (m_bInsideLink) {
34         short fType;
35         if (GetElementType ( strElement, fType)) {
36             ASSERT( HtmlElements[0].fId == HE_A);
37             if (IsContainerEnder( &HtmlElements[0], fType)) {
38                 m_bInsideLink = FALSE;
39                 AddToLinkList();
40             }
41         }
42     }
```

```
43
44 //Perform any special processing associated with the element
45 if (pEf) {
46     // Check whether we should filter this element.
47     DWORD filter = pEf->dwFilter;
48     if (m_dwFilterType & filter) {
49         m_bFilterOutput++;
50         bProceed = TRUE;
51         return TRUE;
52     }
53     /* Process this element. Note that the "this" pointer must be
54     dereferenced in order to give an explicit object context to the
55     element
56     handler processing. The "*pf" represents dereferencing a function
57     pointer
58     in the usual 'C' fashion.
59     */
60     tELEMHANDLER pf = pEf->pfHandler;
61     if (pf)
62         (this->*pf)( pAttrVal, cAttrVals);
63 }
64
65 /* * * Write the tag and associated attributes out to a file * * */
66 //Translate the string into an html element id
67 HTML_ELEMENT* pHe = FindHtmlElement( strElement);
68 /* If it is a block element (vs. an inline), write a line break before
69 and after.
70 Note that if we're inside an anchor element, we will ignore any
71 elements that
72 would ordinarily lead to a block separation, since anchor information,
73 especially
74 the anchor text, must be kept intact inline.
75 */
76 BOOL bIsBlock = FALSE; //Create temporary variable to save result for
77 speed
78 if (pHe && !m_bInsideLink)
79     bIsBlock = !IsDisplayableInlineElement( pHe->fId);
80
81 if (bIsBlock) str = "\n<"; else str = "<";
82 str += strElement;
83
```

```
84     for (i = 0; i < cAttrVals; i++) {
85         string& strAttr = pAttrVal[i]->strAttr;
86         string& strVal = pAttrVal[i]->strVal;
87
88         //Skip any attributes which may have been filtered by the element
89         handler
90         if (strAttr.is_null()) continue;
91         str += " ";
92         str += strAttr;
93
94         if (! strVal.is_null()) {
95             str += "=";
96             str += strVal;
97             str += "\"";
98         }
99     }
100     if (bIsBlock) str += ">\n"; else str += ">";
101     WriteHtmlOutput( str.c_str(), str.length());
102
103     bProceed = TRUE;
104     return TRUE;
105 }
```


Attachment B

